LabTalk/2: A Middleware Approach to HIS Integration

Aamir M. Zakaria, M.D. and Dario A. Giuse, Dr.Ing.
Vanderbilt University Division of Biomedical Informatics, Nashville, Tennessee

LabTalk/2 is an intelligent interface between a legacy order-entry system and a legacy laboratory information system. Unlike other interfaces, LabTalk/2 does more than just transform data from one format to another; it transforms the manner in which data is processed. Utilizing the "middleware" concept, it sits independently between the two systems, decoupling their maintenance needs. Implementation has been successful.

INTRODUCTION

Proper integration of hospital information system (HIS) subcomponents is a major challenge facing medical informaticians. With current network technology, it is no longer acceptable to implement stand-alone system subcomponents which do not communicate with one another. Connectivity and integration eliminate duplication of data entry (for example, patient demographics are required by almost all subcomponents of an HIS) and allow for automated transmission of messages between the subcomponents (for example, transmission of the details of a laboratory test order from an order entry subsystem to a laboratory information subsystem). These benefits come at the price of network hardware and software, interface software, increased complexity of the overall system, and reliance upon network availability.

In 1994, Vanderbilt University Medical Center (VUMC) implemented a legacy* order-entry system. Interfacing that system with the legacy laboratory information system was problematic due to the lack of mapping between the different functions of the two systems. LabTalk/2 was developed as a generic solution to this problem which avoided intervention by vendors of either legacy system. This paper describes the development of LabTalk/2 and presents the results of pilot testing.

BACKGROUND

Challenges to System Integration

System integration requires integration on various levels. These include:

- Data definition
- Functional mapping
- Application level error reporting
- Protocol conversion

Data Definition. Does "blood glucose" on one system represent the same thing on another? An ordering system might define this as a finger-stick, while a lab system might define it as a serum glucose from a venipuncture. These types of data definition inconsistencies must be resolved for successful system integration.

Functional Mapping. Do different systems use data for the same functions? An ordering system uses the same data to generate an order as a laboratory system might use for recording receipt of a specimen. These functional inconsistencies can cause significant problems, as discussed further below.

Error Reporting. How does an application respond to receiving erroneous data? Does it respond in a way which feeds back to the input source in a meaningful way? Is data preserved when errors are generated, or must it be reentered or retransmitted? Is the data source capable of interpreting the error message in a meaningful way, such that the error can be corrected? If the data source is a user at the terminal, and the error message is of the sort, "Please enter a valid 2-digit number", then the error can be easily corrected. However, if the data source is a separate computer system, then this same error message would most likely be meaningless.

Protocol Conversion. Most commercial "integration engines" on the market today act at this level only, converting data from one format to another. While necessary as a component of system integration, protocol conversion alone may not lead to successful integration of complex systems if the other issues are not adequately addressed.

Solutions to System Integration

In some cases, an interface may have already been developed, either by one of the vendors themselves, or by a third-party. If the interface is inadequate, or if no interface exists, extensive customization of one or more of the system subcomponents will most likely be required. This can be a very expensive and time-consuming process.

One ambitious solution is the HERMES project, where an entire architecture was developed for integration.[1] Components are encapsulated and a common message-passing mechanism was developed. An object-oriented database stores information about each component and a broker decides which resources are available to fulfill a particular request.

The Columbia Clinical Repository allows client applications on remote platforms to access a centralized data repository by passing HL7 messages through "Database Access Modules", which then perform SQL queries on a mainframe relational database.[2] A "metadatabase" contains information about routing

*A "legacy" system is defined by William Stead as "a system utilizing internal data definitions and an internal database without regard for external accessibility" (personal communication).

messages to ancillary systems.[3]

MCIS-1 defined an integration architecture which allowed ancillary systems to record data of local interest independently, but stored data of general interest in a central transaction database/hub.[4] These general interest data would be defined according to master data definitions and would be passed to the hub by the ancillary systems over a network. The hub would then be responsible for getting the information to any other system that needed it.

Other projects have focused on other aspects of integration, such as integration of heterogeneous databases[5] and integration of data presentation on a workstation[6,7]. However, little attention has been given to integration between applications which use data in different ways.

VUMC Architecture
VUMC adopted the MCIS-1 plan[4,8], implementing state-of-the-art networking technology to link institution-wide resources to a centralized "Generic Interface". An originating system may pass messages to other systems by simply delivering the data to the Generic Interface and indicating its intended destination. The Generic Interface performs the necessary formatting procedures and guarantees delivery of the message. Routing and formatting details are thus kept transparent to the sender. A relational database (DB2 on an IBM ES/9000 mainframe) acts as a central depository of institution-wide information.

The Current VUMC HIS to LIS Interface
VUMC has a core HIS developed by SMS (Malvern, PA) and a Laboratory Information System (LIS) manufactured by CHC (Houston, TX). Admission/Discharge/Transfer information is transmitted to the LIS through a customized SMS interface. This interface resides on a minicomputer independent of both the HIS and the LIS, and can be thought of as a type of middleware which reformats and performs protocol conversion. When the SMS Order-Entry subsystem was brought on-line, orders for lab tests were also transmitted through the SMS interface. There was some discussion about asking CHC to create a customized orders queue to improve the transfer of data between the two systems. This was not done because the lifespan of the LIS was uncertain, and investment in customized nonreusable code did not appear judicious.

In general terms, several problems were caused by the interface between two systems that are oriented around different tasks: the order-entry system is oriented around orders, while the LIS is oriented around specimen receiving ("accessioning"). Although the lab orders data was being transmitted in the proper format by the SMS interface, the intended *function* of the data had changed from being an order (a request for service) to being interpreted and handled by the lab system as a record of specimen receipt.

In practical terms, this meant that the lab computer was receiving a dump of orders data which it had to hold onto indefinitely. Not infrequently, lab tests which had been ordered were not actually collected (e.g. duplicate or cancelled orders); these data items accumulated on the lab computer, taking up valuable disk space unnecessarily. When a specimen did arrive in the lab, the lab technician's task was to manually sift through all the lab orders for that patient to find the one which matched the specimen. This was time-consuming and error-prone.

Assignment of sample numbers was also a problem. The lab computer system assigns a sample number serially for each specimen received. When the order-entry system was used to feed the lab system, each order, not just each specimen, received a specimen number at the time the order was generated. However, one of the major laboratory subdepartments, Microbiology, relies upon sequential sample numbers assigned at the time of specimen receipt. Using orders data for specimen receipt functions caused a major problem. In this case, the mismatch was such that Microbiology could not interface with the order-entry system at all, forcing duplicate entry of data in the lab.

METHODS

Middleware is a possible solution for this type of integration need. Middleware is software (and hardware) that "sits between" the different systems, negotiating a mapping from one to another, utilizing input and output mechanisms which are already present, avoiding the need to customize vendor software or reorganize existing architecture. LabTalk/2 is an experiment in middleware. The task of middleware can be as simple as translating messages from one format to another. Often, however, because different systems are oriented around different types of work, a more complex mapping of both format and function is necessary. This was the case with the VUMC orders and lab systems.

Design Considerations
Client-Server. Our original prototype did not incorporate a client-server configuration. A single connection was made from a LabTalk/2 workstation to the mainframe. Because of the complexities of configuring and maintaining such a connection, it was felt that a client-server architecture would be preferable to allow a single PC-mainframe connection to serve multiple PC clients (with a second backup server also configured, in the event of failure of the primary server).

Database. We initially planned on storing the orders data on the LabTalk/2 Server machine. However, at VUMC, the Generic Interface already redundantly stores orders data as they are generated by the HIS in a relational database on the IBM ES/9000 mainframe. In order to avoid updating and synchronization problems, we decided not to create a separate orders database. This plan created a greater degree of dependence on the mainframe, and hence a greater degree of vulnerability to mainframe downtime. There are plans to eventually shadow the database elsewhere, which will diminish this vulnerability.

Two new columns were added to the DB2 Orders table: a date/time stamp to record the time an order was transmitted to the lab computer, and a counter to record the number of times an order was transmitted. The latter field was created because situations could potentially arise where an order would need to be transmitted multiple times. For example, the lab might request an order by mistake, or a specimen container may break in the lab and need to be recollected. If a known mistake occurs, or a specimen needs to be recollected, the lab may "uncollect" the specimen in LabTalk/2, resetting those fields in the database. Quality control reports will be generated weekly to ensure that orders are not being retransmitted excessively.

Communication Between LabTalk/2 and the Lab Computer. The initial design called for a direct communication link between LabTalk/2 and the lab computer. After some discussion, it was felt that this would violate one of the principles of the present VUMC architecture, which places the Generic Interface at the center of most data transfers between system subcomponents. A small amount of data transfer does take place directly from the lab computer to the LabTalk/2 Server - a weekly transfer of two configuration files via ftp.

System Architecture
System components for LabTalk/2 include a proprietary Order Entry System (SMS), a VUMC "Generic Interface Engine" (GIE), a relational database (DB2), a proprietary Laboratory Information System (CHC), one or more LabTalk/2 Specimen Receiving Clients, and a LabTalk/2 Communications Server (see Figure 1).

A client-server architecture is used, as has been described previously for other patient-care applications.[1,7,9,10,11]. In this model, a client composes a request, which is passed to the server. The server fulfills the request and passes the results back to the client. Multiple layers of client-server interactions are used in LabTalk/2. The LabTalk/2 Communications Server is itself a client of the GIE; the GIE, in turn, is a client of the database.

The system is intended to work in the following way. Orders are entered into the Order Entry System and stored in the database. Barcoded requisition slips are printed on the ward and sent to the lab with the specimen upon collection (these processes have been in place for a number of years).

Upon receipt of the specimen in the lab, the LabTalk/2 Specimen Receiving Client is started up (if it is not already running). The client runs on an IBM-compatible PC workstation as a Visual REXX executable in the OS/2 operating system. The LabTalk/2 Client sends a startup message to the LabTalk/2 Communications Server. When a client connects, the server initiates an LU6.2 APPC conversation with the Generic Interface Engine sitting on the IBM ES/9000 Mainframe. A separate conversation thread is initiated with the GIE for each LabTalk/2 Client which connects with the LabTalk/2 Server (initial implementation includes 11 client machines).

Identification information from the requisition slips is

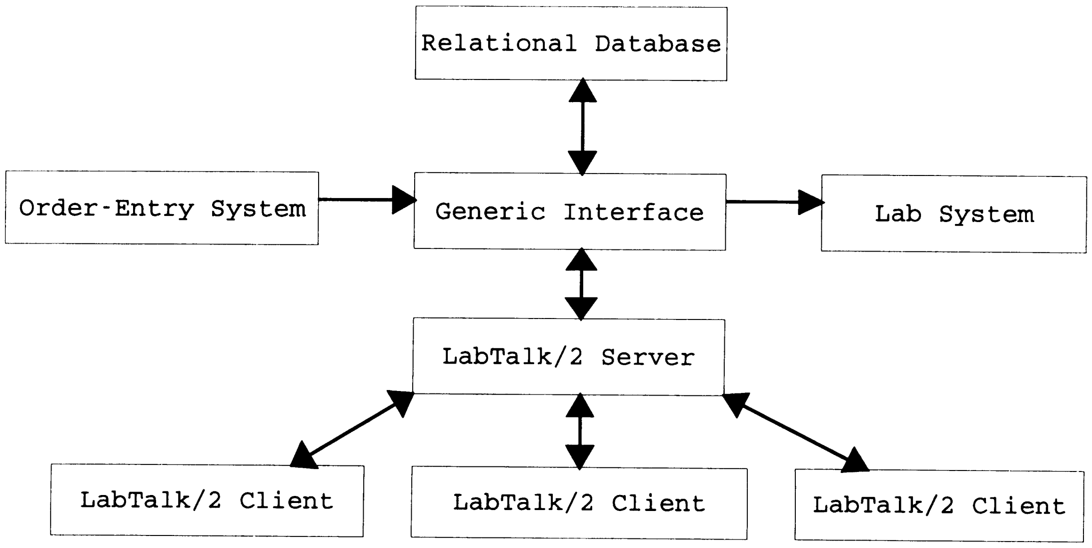

Figure 1. LabTalk/2 Architecture Overview.
Orders are passed to the Generic Interface and stored in the database. When specimens are received in the lab, orders data are requested by LabTalk/2 via the Generic Interface. After verification, the Generic Interface is requested to format the selected orders and transmit them to the lab computer.

```
                              LAB RECONCILIATION REPORT
                               7 NORTH 4/4/95 04:00
According to the laboratory computer, the following lab orders had not been received as of Midnight
(4/4/95 00:00).  If these orders are not going to be collected, please d/c them in VCARE.

Room   MRN              Name             Order  Occ     Test          Date     Time     Pri
< 24 hours late
7001   02312342-1       Richard Smith    723-    1      Urinalysis    4/3/95   12:00    R
                                         (Note: Last Urinalysis on this pt received in lab 4/3/95 13:05)
                                         411-    1      ABG           4/3/95   16:00    R
7004   02323338-2       John Jones         3-    1      Blood Cult    4/3/95   20:30    R
24-48 hours late
48-72 hours late
7001   02312342-1       Richard Smith    701-    1      CHEM-7        4/1/95   12:00    S
                                         (Note: Last CHEM-7 on this pt received in lab 4/1/95 07:00)
> 72 hours late
```

Figure 2. Sample LabTalk/2 Reconciliation Report.
VCARE refers to the hospital order-entry system. Patient names in this illustration are fictitious.

read with a barcode reader by the LabTalk/2 Specimen Receiving Client, utilizing a Visual REXX GUI. This information (medical record number and case/episode number) is passed to the LabTalk/2 Communications Server via an IPX named pipe (a NetWare communications protocol), and then to the Generic Interface via APPC (an IBM communications protocol). The Generic Interface retrieves the name and location of the patient from the DB2 database and transmits them back to the Client for verification. If verified by the lab technician, the order number is then scanned in by the barcode reader. Multiple order numbers can be scanned consecutively for the same patient.

For each order number, a database lookup is performed via the same mechanism as just described for demographics. If an order does not exist in the database, an error message is passed back to the client and displayed to the user. When all order numbers have been entered, and the user is satisfied with the list, the user clicks a 'Done' button on the GUI, at which point a request is sent to the Generic Interface to send those orders data to the lab computer. The Generic Interface performs the appropriate formatting and transmits the orders data to the lab computer via a TCP/IP socket interface. A date/time stamp is recorded in the database indicating that the sample has been "accessioned" by the lab and the orders data has been transmitted to the lab computer. The lab computer processes the request, assigns a sample number, and prints labels for the specimen(s).

A report has been designed which will print at approximately 4:00 every morning on the nursing stations (see Figure 2). This report queries the database looking for active lab orders with a date and time of desired collection which is past due (i.e., more than 4 hours beyond scheduled collection). These items could represent either uncollected specimens, duplicate orders, or inappropriate orders (e.g., urinalysis on an anuric patient). To help the nurse decide what to do with these orders, a note is made of the last time a similar specimen was received in the lab (for example, if there is a CBC order still pending from 2 days ago, but a separate CBC specimen was received in the lab yesterday, the outstanding order is probably no longer necessary). Duplicate and inappropriate orders need to be discontinued in the order-entry system, so that the database accurately reflects only active orders which still need to be collected. Accuracy of this database will become even more important if it becomes a resource for billing purposes (obviously, patients should not be charged for duplicate or inappropriate orders). Because the order-entry system is not presently implemented hospital-wide, the reconciliation report loses effectiveness if a patient is transferred from an implemented to a non-implemented unit, because the nurses on the non-implemented units do not know how to discontinue orders in the system.

PILOT RESULTS

As an initial evaluation of the feasibility of LabTalk/2, we developed a prototype and measured response times for the "client". This prototype "client" communicated directly with the Generic Interface, without an intervening "server". When a request for demographics or orders data was about to be sent to the Generic Interface, an internal elapsed time clock was reset. When the data was received back from the Generic Interface, the clock was stopped. This elapsed time therefore represented the time to transmit a request from LabTalk/2 to the Generic Interface, perform an SQL query on the database, and transmit the results back to LabTalk/2. For a sample of 20 demographic and order requests, using a fully populated version of the database, the average response time was 240 milliseconds, with no significant difference between demographic lookups and order lookups. We felt this was adequate, and we proceeded with development of the client-server version of LabTalk/2.

We then repeated the timing studies on the client-server version, with the client and server residing on separate workstations. These response times would necessarily be higher because of the intervening layer of client-server communication. This time, the average response time was 1.1 seconds.

DISCUSSION

LabTalk/2 enables two HIS subcomponents to communicate more successfully with each other by effecting a transformation of data function (as well as format), to correspond with the differing functions of the data on the each system. From the lab's viewpoint, LabTalk/2 is more oriented towards specimen receiving than the previous interface. Orders that are "pending", i.e., orders for which a sample has not yet been received by the lab, no longer appear on the lab side, and hence do not distract the technicians from processing specimens which have been received. This also solves the problem with orders for which a specimen is never actually obtained: because LabTalk/2 is driven by the receipt of specimens, such orders never need to be seen, and need not be disposed of in the lab. The lab computer is no longer burdened with unnecessary orders data; only those data corresponding to received specimens are transferred. Laboratory technicians no longer need to manually sort through lab orders when they receive a specimen; a specific request is sent to the database for retrieval of the orders data corresponding to the specimens received.

By delaying order transmission until specimen receipt, sample numbers may be assigned sequentially. This has now allowed Microbiology to reap the advantages of automated data transfer from the order-entry system.

Intelligence can be applied to the handling of specimens for which orders have been discontinued: LabTalk/2 can perform a query on the database to look for any outstanding orders for the same lab test ordered for collection within a 30 minute time window of the original order. Occasionally, duplicate tests are ordered, then one of the duplicates is cancelled; however, the wrong requisition is destroyed. This logic can potentially allow an easy resolution for this situation. Note that the laboratory is currently not using this feature because their personnel do not feel comfortable processing an order for which they do not have a paper requisition.

The response times with the client-server version of LabTalk/2 were somewhat slower than we would have liked, but still well within acceptable limits for practical use. We feel that we might be able to improve upon these times by rewriting the client transport routines in C instead of REXX.

LabTalk/2 is entirely transparent to both SMS and CHC systems - no reengineering was required. This amounted to significant savings in cost - all new code was written in-house, leveraging the Generic Interface and DB2 repository that had been implemented as part of the VUMC architecture, avoiding the more expensive and time-consuming process of vendor-based custom programming. The time required was only four months from design to implementation (1.5 FTEs). Implementation of LabTalk/2 enabled the elimination of the previous MicroVax interface, with all its associated maintenance costs.

By remaining sufficiently generic, i.e. avoiding implementation details of the specific systems being interfaced, LabTalk/2 embodies the concept of "plug-compatible programming"[12], enhancing its reusability. If either system is ever replaced or changed, LabTalk/2 can still perform as a generic orders-system-to-lab-system interface, leaving the Generic Interface to worry about reformatting and protocol conversion. No changes would be necessary in LabTalk/2, only in the Generic Interface.

CURRENT AND FUTURE STATUS

LabTalk/2 went "live" at the end of June, 1995, and has been successfully running ever since. It has been enthusiastically accepted by the laboratory personnel and is felt to greatly improve the workflow in the lab. We are now in the process of formally evaluating the impact of LabTalk/2 on our organization. We expect to extend the concept to integration of the orders system with the pharmacy system.

CONCLUSION

This experiment shows that the middleware concept is technically feasible and can be successfully implemented in a large organization. Future capabilities include tracking of laboratory samples and task lists for specimen collection. The potential of the system to increase productivity, throughput, and quality in processing laboratory specimens is substantial.

Acknowledgements

This work was supported in part by IAIMS grant G08-LM05443-03 from the National Library of Medicine.
We wish to acknowledge Randall Bates for his programming efforts on the Generic Interface, Nancy Proctor as a liaison with the Information Management Department, and Susan Steane as a liaison with the Laboratory.

References

1. Van Mulligen E, Timmers T. Beyond clients and servers. In: Ozbolt JG, ed. *Proceedings of the 18th Annual Symposium on Computer Applications in Medical Care.* Philadelphia: Hanley & Belfus, Inc. 1994: 546-550.
2. Johnson SB, Hripcsak G, Chen J, Clayton P. Accessing the Columbia Clinical Repository. In: Ozbolt JG, ed. *Proceedings of the 18th Annual Symposium on Computer Applications in Medical Care.* Philadelphia: Hanley & Belfus, Inc. 1994: 281-285.
3. Johnson SB, Cimino JJ, Friedman C, Hripcsak G,

Clayton PD. Using metadata to integrate medical knowledge in a clinical information system. In: Miller RA, ed. *Proceedings of the 14th Annual Symposium on Computer Applications in Medical Care*. Washington: IEEE Computer Society Press. 1990: 340-344.

4. Stead WW, Borden RB, Boyarsky W, et al. A system's architecture which dissociates management of shared data and end-user function. In: Clayton PD, ed. *Proceedings of the 15th Annual Symposium on Computer Applications in Medical Care*. New York: McGraw-Hill, Inc. 1992: 475-480.
5. Marrs KA, Steib SA, Kahn MG. Unifying heterogeneous distributed clinical data in a relational database. In: Safran C, ed. *Proceedings of the 17th Annual Symposium on Computer Applications in Medical Care*. New York: McGraw-Hill, Inc. 1993: 644-648.
6. Cornet R, Van Mulligen EM, Timmers T. A cooperative model for integration in a cardiology outpatient clinic. In: Ozbolt JG, ed. *Proceedings of the 18th Annual Symposium on Computer Applications in Medical Care*. Philadelphia: Hanley & Belfus, Inc. 1994: 590-594.
7. Young CY, Tang PC, Annevelink J. An open systems architecture for development of a physician's workstation. In: Clayton PD, ed. *Proceedings of the 15th Annual Symposium on Computer Applications in Medical Care*. New York: McGraw-Hill, Inc. 1992: 491-495.
8. Stead WW, Sittig DF. Integrated Advanced Information Management Systems: strategies and architectures for fast track development. In Press: Prokosh HU, Dudeck J, eds. *Hospital Information Systems: Design and Development Characteristics, Impact and Future Architectures*. Amsterdam: Elsevier Science B.V. 1995.
9. Chueh HC, Barnett GO. Client-server, distributed database strategies in a health-care record system for a homeless population. Journal of the American Medical Informatics Association. 1994; 1:186-198.
10. Van Mulligen EM, Timmers T, Van Bemmel JH. A new architecture for integration of heterogeneous software components. Methods of Information in Medicine. 1993; 32:292-301.
11. Huff SM, Haug PJ, Stebens LE, Dupont RC, Pryor TA. HELP the next generation: a new client-server architecture. In: Ozbolt JG, ed. *Proceedings of the 18th Annual Symposium on Computer Applications in Medical Care*. Philadelphia: Hanley & Belfus, Inc. 1994:271-275.
12. Greenes RA. Promoting productivity by propagating the practice of plug-compatible programming. In: Miller RA, ed. *Proceedings of the 14th Annual Symposium on Computer Applications in Medical Care*. Washington: IEEE Computer Society Press. 1990: 22-26.